2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 24-25 December, Dhaka

Android Apps Success Prediction Before Uploading on Google Play Store

Golam Md. Muradul Bashir[1], Md. Showrov Hossen[2], Dip Karmoker[3], and Md. Junaeed Kamal[4]
Faculty of Computer Science and Engineering
Patuakhali Science and Technology University
Patuakhali, Bangladesh
murad98csekuet@yahoo.com[1] , sourovhossen96@gmail.com[2] , karmokerdip75@gmail.com[3] and aanik454@gmail.com[4]

Abstract— **Nowadays among all the mobile apps distribution platform Google Play Store is one of the most attractive and user effective platform for android apps. Every day this platform gets drenched with a huge amount of new android apps. Developers are always trying to make successful their new apps through their skilled independent work or team work. This platform is getting competitive day by day for the developers to retain their place in the market. But this growing competitive sector can be easier for the developers if they can determine the new app's success before uploading it on Google Play Store. As we know that usually an app's success is determined through the app's user rating and installation number, our work relates with the purpose to assume the new app's success through the prediction of user rating and installation number before uploading on the Google Play Store.**

Keywords—android, mobile apps, rating, installation

I. INTRODUCTION

The Google Play Store is one of the largest app market among all the mobile apps distribution platform in the world. With the ever growing mobile app market android app developers are rising. To keep pace with the competitive android app market an effective way should be determined by the developers to make their current position stable to retain their place in the market and it may be a solution for the developers to predict the success through the rating of end users and installation as we know that the success of a new app is determined by it's user rating and installation number. Attempting the research on it datasets are scraped from the Google Play Store. Recently, a lot of researches have worked on this data as we have seen in paper [1] [2] [3].The scraped dataset that we have used, contains 267000 unique apps data. Analyzing the dataset we have created several Bar Charts to visualize the relationship between different attributes of it such as category, rating, size, installs, types etc. We have used some machine learning algorithms like as Random Forest, K- Nearest Neighbor and Support Vector Machine to assume the success of the new apps.

II. MOTIVATION

Analyzing some android apps related research papers, we are motivated to make a research on android apps success prediction. We have observed some important issues, haven't analyzed yet:

- Before launching any new app on Google Play Store predicting the most accurate installation number and user rating.
- Help developers to gather predictive knowledge about their success for the new apps before uploading on Google Play Store so that they can modify the apps if it needs.

III. LITERATURE REVIEW

In paper [1], the authors tried to represent the review-rating mismatch, through establishing multiple systems that can automatically detect the inconsistency between these two, to prove this mismatch they implemented two machine learning approaches, it used different classifiers such as Naïve Bayes Classifier, Decision tree, Decision table, Decision stump, and few other algorithms, and the other approach focused on deep learning techniques. They also hosted several surveys in order to learn the thoughts of users and developers regarding this mismatch. The outcome of the survey was quite expected, both the Developer and end users of Android app agreed that rating of an app should match with its corresponding review and they also asserted to have an automated system to detect the mismatch between rating and review if there is any. This paper proposed a wonderful way to represent the review-rating mismatch, we got inspired by their work and gave an effort to create a feature based on the review and rating mismatch and eliminate those which has a higher difference in mismatch but due to time constraint three of our members could manually annotate only 2000 reviews out of 199763 [1] individually, due to this huge variance we could not get a proper result and dropped the idea of doing so.

Similarly, In paper [3], the authors observed that how store-ratings after reaching a certain value does not affect the overall store rating even after users rate it, they also noticed that when an app is updated their rating varies version wise, but it is not observable in the store-rating. Therefore, they came up with the idea of version rating which they could calculate based on the calculation of store rating which was available in App Store. This idea was proposed in order to help developers gain incentive to develop apps even after the store rating reaches a threshold value and they recommended that every App store owner should display the current version rating. Their approach of calculating version rating to make developers work for the updates is very efficient.

Another paper working on the same aspect [4], the author states that the numeric rating as in the stars given by users have a huge difference than compared to the reviews given by them thus a rating system has been proposed by the author which will remove the ambiguity created by the mismatch of the rating and respective review by the same user. It has been seen that how much we as users are dependent on others opinion while taking any decision so according to the author people install the app based on the rating that is given for that particular app. According to the author, the ambiguity and the biasness to the summarized rating of the users are two sub-problems. Further explaining the problem, he added that earlier people used to only extract the rating through the comments instead of the star rating associated with it. To solve the problems, he proposed a system that will initially conduct a sentimental analysis on the reviews of end users. Then from the polarity it will generate a numeric rating. Thus, the average of the rating from the sentiment analysis and the star ratings of end users will generate a final rating. This proposal would reduce the confusion of users and allow them to have a final rating based on both review and star rating. This paper shows a strong relationship between the user rating and the reviews, which helped us of picking up the idea of using the reviews of the app in our work and we did a thorough research on the reviews that are explained in later sections of this paper.

Correspondingly, in paper [3], Luiz et al. proposed a framework for mobile application developers with which they will be able to bring in modification on features those are found negative based on the end users' evaluation of their application. Their Framework was designed to make developers realize that sentiment rating provides a more accurate value of user feedback for an application than star rating and how important it is to take account of the biasness of the features that affect the overall rating of an application. This paper guided us to find the sentiment mean by showing how it gives a more accurate value than star rating. We simply used a python library for finding the Sentiment mean rather than using their strategy.

Considering the fact that how important polarity values are, thus from Fu, Lin, and Li work [5], it helped us to get the idea of removing the inconsistent reviews that will basically reduce noise from the dataset and give better performance in sentiment analysis, and therefore will generate polarity value more accurately. Fu, Lin, and Li proposed WisCom, a system that is able to analyze a vast number of user ratings and comments in the mobile application markets in three different levels and the number of this ratings and comments at least ten millions.

In paper [6], the author performed an exploratory data analysis on their collected App Store data, discovered relationships with some appointed features. With the help of these extracted features and the recent reviews of user they tried to predict the success of an app after it is launched into the Google Play Store.

All the papers mentioned above helped us to get all the ideas that we incorporated in our work, which made our work more organized and eventually helped us to produce a better analysis of the features that are present in our dataset.

IV. System Architecture

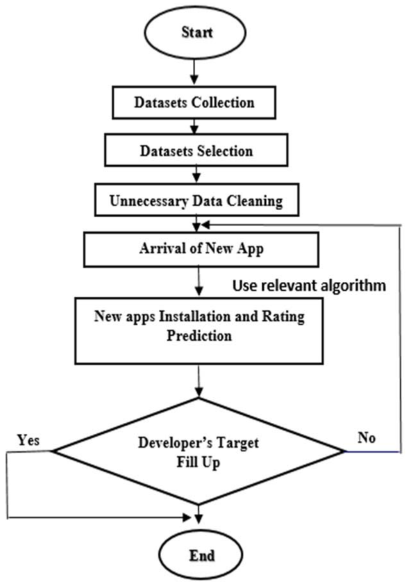

Start
Datasets Collection
Datasets Selection
Unnecessary Data Cleaning
Arrival of New App
Use relevant algorithm
New apps Installation and Rating Prediction
Developer's Target Fill Up
Yes
No
End

Fig. 1. Methodology Flowchart

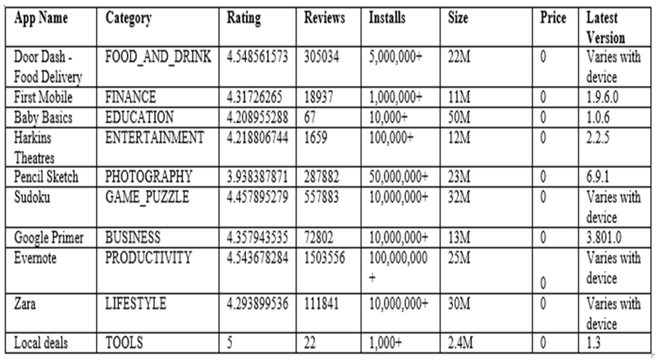
In this research work our methodology is proceeded after scraping data set of Google Play Store. Then we will select the valid data and clean the garbage values of the dataset. After that when the developer brings a new app, using the relevant algorithm with the dataset we will be able to predict the new app's user rating. After that when the developer brings a new app, using the relevant algorithm with the dataset we will be able to predict the new app's user rating.

V. Data Preprocessing

Most of the time data comes with garbage values, which should be handled before it affects the performance of trained models that predict the outcome. Various steps are used to preprocess that data.

A. Datasets

In our experimental evolution here we considered a collection of Google Play Store dataset. The dataset contains the columns App Name, Category, Rating, Reviews, Installs, Size, Price, Content Rating.



App Name	Category	Rating	Reviews	Installs	Size	Price	Latest Version
Door Dash - Food Delivery	FOOD_AND_DRINK	4.548561573	305034	5,000,000+	22M	0	Varies with device
First Mobile	FINANCE	4.31726265	18937	1,000,000+	11M	0	1.9.6.0
Baby Basics	EDUCATION	4.208955288	67	10,000+	50M	0	1.0.6
Harkins Theatres	ENTERTAINMENT	4.218806744	1659	100,000+	12M	0	2.2.5
Pencil Sketch	PHOTOGRAPHY	3.938387871	287882	50,000,000+	23M	0	6.9.1
Sudoku	GAME_PUZZLE	4.457895279	557883	10,000,000+	32M	0	Varies with device
Google Primer	BUSINESS	4.357943535	72802	10,000,000+	13M	0	3.801.0
Evernote	PRODUCTIVITY	4.543678284	1503556	100,000,000 +	25M	0	Varies with device
Zara	LIFESTYLE	4.293899536	111841	10,000,000+	30M	0	Varies with device
Local deals	TOOLS	5	22	1,000+	2.4M	0	1.3

Fig. 2. Sample Dataset

B. Data Cleaning

The raw data are in different formats with garbage values that should be converted into a similar format to use data efficiently in building machine learning model.

C. *Conversion of Data into Appropriate Forms*

Size: For example, the size of the app is in “string” format. We have to convert it into a numeric value. If the size is “22M”, to get the numeric value of ‘22’ then ‘M’ was removed. If the size is “273k”, which depicts app size in kilobytes, then first ‘k’ should be removed and the size should be converted to an equivalent of ‘megabytes’.

Installs: The installation values are in “string” format. This contains numeric values with commas. The commas should be removed. And also, the ‘+’ sign should be removed from the end of each string.

Category and Content Rating: The Category and Content Rating consist of categorical values and it should be converted to numeric values if we need to perform regression.

Price: The data of price is provided in “string” format. The dollar sign should be removed from the string to convert the data into numeric form.

Analyzing the data, ratings of the app can be concluded as the most significant parameter that plays a vital role in depicting how better the app performs compared to the other apps in the market. It also indicates how well the company works on implementation of the feedback provided by the end users. After all, users are the key to modern software businesses [7].

VI. Experimental Evolution

A. *Category vs App*

The Category column in our dataset has 34 different types of categories. Here we have shown the number of different types of apps against the categories. Figure 3 shows the Bar Chart of the number of categories.

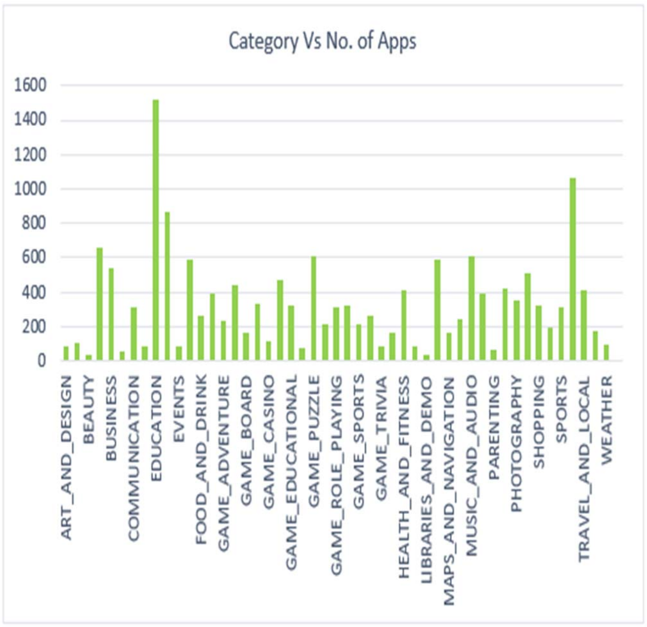


Fig. 3. Bar chart of category against number of apps

TABLE I : *Top 10 most dominant categories by number of apps*

Place	App	Frequency
1	Game	35508
2	Education	33394
3	Tools	21592
4	Books and reference	21377
5	Entertainment	20604
6	Music and audio	17876
7	Personalization	15044
8	Lifestyle	10534
9	Finance	10342
10	Business	10230

B. *Category vs Installs*

Here a Bar Chart is created to visualize the installation numbers of the apps. Figure 4 shows the Bar Chart of all category against the number of Install.

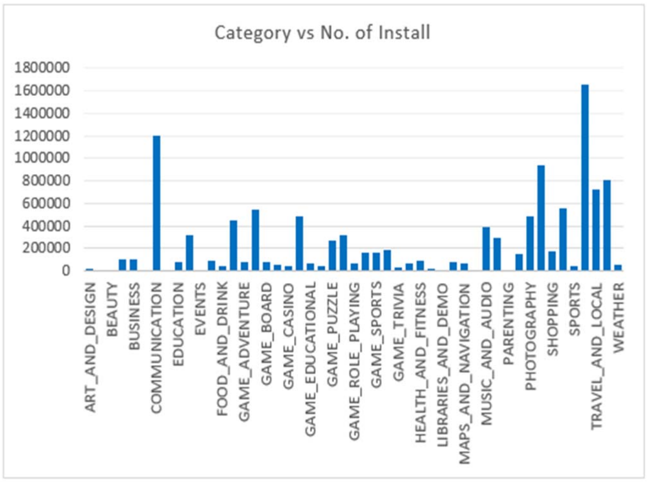


Fig. 4. Bar Chart of all category against the number of install

C. *Category vs Average Rating*

The average rating of apps are presented with a Bar Chart. Figure 5 shows the Bar Chart of all category against the Average Rating.

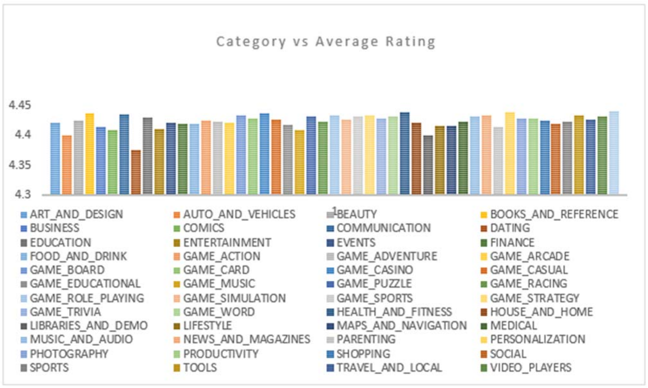


Fig. 5. Bar chart of category against number of average rating

VII. Result Analysis

A. Result Through Used Algorithms

Random Forest: Random forest is a tree based algorithm which is used to build various decision trees and make the combination of their output to improve deduction ability of the model. Pseudo code of Random Forest algorithm is

- Select "k" features randomly from total "m" features (Category, Size, Price) Where k << m
- Calculate the node "d" among the "k" features using the best split point.
- Split the node into daughter nodes using the best split.
- Until "l" number of nodes has been reached, repeat 1st to 3rd steps.
- Repeat steps 1st to 4th for "n" number times to create "n" number of trees to build the forest.

TABLE II : Predicted result of user rating by Random Forest algorithm

App Name	Original Rating	Predicted Rating
Angry Birds	4.5	4.41
Facebook	4.1	4.22
Messenger	4.36	4.33
UC Browser	4.5	4.31
Clean Master Lite	4.58	4.36

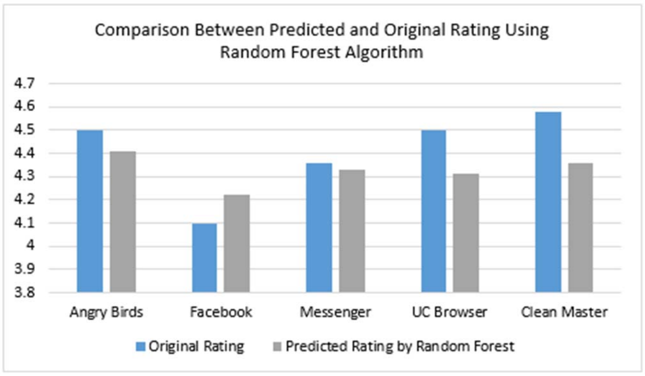


Fig. 6. Bar chart of comparison of original rating and predicted rating using Random Forest Algorithm.

TABLE III : Predicted result of installation number by Random Forest algorithm

App Name	Original Installation No.	Predicted Installation No.
Angry Birds	10000000+	11984169
Facebook	1000000000+	78845910
Messenger	500000000+	164585359
UC Browser	500000000+	37219510
Clean Master Lite	50000000+	17226013

KNN: An unknown sample is classified by the KNN (k-nearest neighbor) algorithm based on it's neighbors known classification. In this algorithm, when it is a sample of unknown classification, then it could be predicted by considering the classification of its nearest neighbor samples. Pseudo code of KNN is

- Classify(X, Y, x) // X:training data, Y:class labels of X, x: unknown sample
- for i =1 to m do
- Compute distance d(Xi, x)
- end for
- Compute set I containing indices for the k smallest distances d(Xi, x)
- Return majority label for{Yi where i ∈ I}

TABLE IV: Predicted result of user rating by K-Nearest Neighbor algorithm

App Name	Original Rating	Predicted Rating
Angry Birds	4.5	4.61
Facebook	4.1	4.05
Messenger	4.36	4.43
UC Browser	4.5	4.37
Clean Master Lite	4.58	4.68

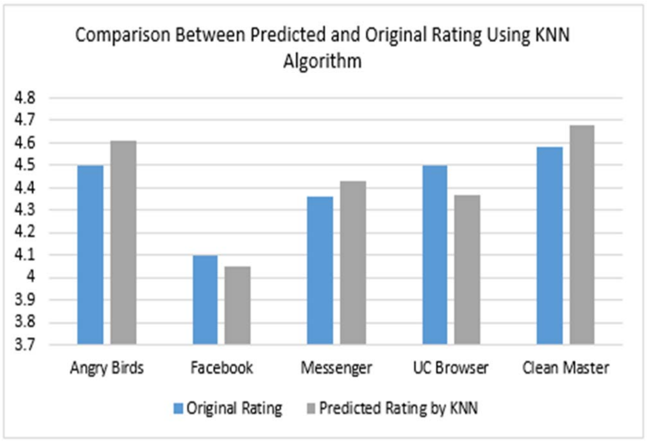


Fig. 7. Bar chart of comparison of original rating and predicted rating using K- Nearest Neighbor Algorithm.

TABLE V: Predicted result of installation number by K-Nearest Neighbor algorithm

App Name	Original Installation No.	Predicted Installation No.
Angry Birds	10000000+	10129323
Facebook	1000000000+	455465897
Messenger	500000000+	359823985
UC Browser	500000000+	187489232
Clean Master Lite	50000000+	56673210

SVM: Reducing the misclassification error a machine learning algorithm usually tries to find a boundary in case of linearly separable data in two dimensions, when that divides the data in such a way that can be. In SVM (support vector machine) algorithm, the decision boundary is chosen and the distance from the nearest data points of all the classes is maximized by it. It defines the most favorable decision boundary.

When it is nonlinearly separable data, the simple SVM algorithm doesn't work. Then Kernel SVM, is used. Pseudo code of SVM is

- Initialize *k* solutions
- Call SVM algorithm to evaluate k solutions
- T= Sort (S_I……S_k)
  While classification accuracy != 100 or number of iteration != 10 do
- for i= 1 to m do
- select S according to its weight
- sample selected S
- store newly generated solutions
- call SVM algorithm to evaluate newly generated solutions
- end for
- T= Best (Sort $S_{I,\ldots}S_k$ + m),k)
- end while

TABLE VI: Predicted result of user rating by Support Vector Machine algorithm

App Name	Original Rating	Predicted Rating
Angry Birds	4.5	4.54
Facebook	4.1	4.09
Messenger	4.36	4.40
UC Browser	4.5	4.44
Clean Master Lite	4.58	4.55

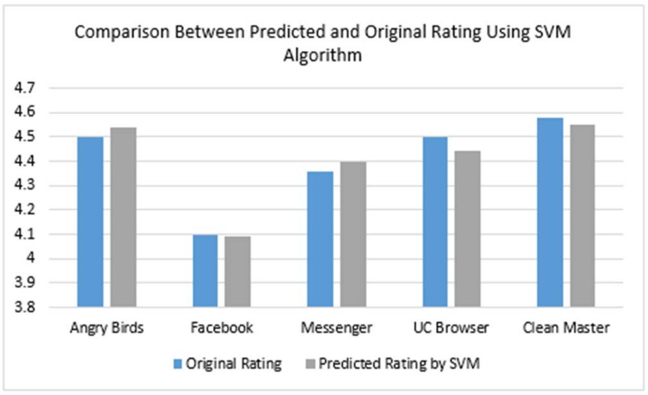


Fig. 7. Bar chart of comparison of original rating and predicted rating using SVM Algorithm.

TABLE VII: Predicted result of installation number by Support Vector Machine algorithm

App Name	Original Installation No.	Predicted Installation No.
Angry Birds	10000000+	10009323
Facebook	1000000000+	975465897
Messenger	500000000+	634324990
UC Browser	500000000+	298753459
Clean Master Lite	50000000+	43467023

B. *Success Prediction*

The formula we used to defining the success is, when the average rating and install count being greater than or equal to developer's defined target value.

Success = 1 if Rating > Target Value
0 else
Success = 1 if Installation > Target Value
0 else

VIII. Conclusion And Future Work

The Google Play Store is the largest app market in the world. In this work, we have introduced a new framework that provides the mobile app developers an effective way to successfully explore in competitive mobile application market. This framework helps the developers to predict the new app's success through assuming the installation rate and user rating. After extracting various Google Play Store data, here some machine learning algorithms are used to predict the success. We have made the comparison of predicted rating and installation number with the original rating and the installation number. According to the comparison of original and predicted values of rating and installation using different algorithms we have come to a decision that the KNN and SVM algorithm can predict the success rate more accurately than the Random Forest. Our proposed concept will facilitate the developers to assume that whether he is proceeding in the right way or not. In future we will predict of the type of reviews by using the regression model and solve the review, rating mismatch. Identify the categories and stats of the most installed apps. Explore the correlation between the sizes of the apps, the version of android on the number of installs. Define user dissatisfaction with a new update.

References

[1] R. S. G. G. N. a. M. S. Aralikatte, "Fault in your stars: an analysis of android app reviews," in *ACM India Joint International Conference on Data Science and Management of Data*, Goa,India, 2018.

[2] I. J. M. N. M. A. B. B. T. D. S. a. H. A. E. Ruiz, " Examining the rating system used in mobile-app stores," *IEEE Software,* vol. 33, no. 6, p. 86– 92, 2016.

[3] W. V. F. A. R. M. F. S. T. C. D. G. M. A. a. R. L. Luiz, "A feature-oriented sentiment rating for mobile app reviews," in *World Wide Web Conference*, Lyon,France, 2018.

[4] M. R. Islam, "Numeric rating of apps on google play store by sentiment analysis on user reviews," in *International Conference on Electrical Engineering and Information Communication Technology*, Dhaka, Bangladesh, 2014.

[5] B. L. J. L. L. F. C. H. J. a. S. N. Fu, " Why people hate your app: Making sense of user feedback in a mobile app store," in *19th ACM SIGKDD international conference on Knowledge discovery and data mining*, California, Berkeley, 2013.

[6] K. A. ,. I. W. I. Abdul Mueez, "Exploratory Data Analysis and Success Prediction of Google Play Store Apps," BRAC University, Dhaka,Bangladesh, 2018.

[7] A. Thakre, "Medium," The Research Nest, 24 March 2018. [Online]. Available: https://medium.com/the-research-nest/data-science-tutorial-analysis-of-the-google-play-store-dataset-c720330d4903. [Accessed 19 May 2019].